Lecture 5 – Maths for Computer Science
Structured Graphs (part 2)

Denis TRYSTRAM
Lecture notes MoSIG1

november 2021

Objectives

Review some structured graphs and study their properties.

- Complete graphs
- Cycles
- Meshes and torus
- Hypercubes
- Trees

Complete graphs (or cliques)

Definition.
Each vertex of K_n is connected to all the other vertices.

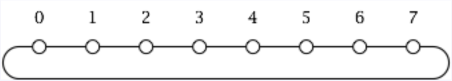
- Connected ($D = 1$)
- Regular graph ($\delta = n - 1$)
- Number of edges $\Sigma_{1 \leq k \leq n-1}\ k = \frac{(n)(n-1)}{2}$

Rings or Cycles

Definition.
Each vertex of C_n has exactly one predecessor and one successor.

Coding of edges

$$\{\{i,\ i+1 \bmod n\}\ \mid\ i \in \{0,\ 1,\ \ldots,\ n-1\}\}.$$



- Connected ($D = \lfloor \frac{n}{2} \rfloor$)
- Regular graph ($\delta = 2$)
- Number of edges n

An interesting observation

Proposition.

If every vertex of a graph G has degree ≥ 2, then G contains a cycle.

An interesting observation

Proposition.

If every vertex of a graph G has degree ≥ 2, then G contains a cycle.

Proof

Let us assume by contradiction that we have a cycle-free graph G all of whose vertices have degree ≥ 2.

Let us view graph G as a park where every vertex of G is a statue, and every edge is a path between two statues.
The fact that every vertex of G has degree ≥ 2 means that if we take a stroll through G, then every time we leave a vertex $v \in V$, we can use a *different* edge/path than we used when we came to v.

Meshes and Torus

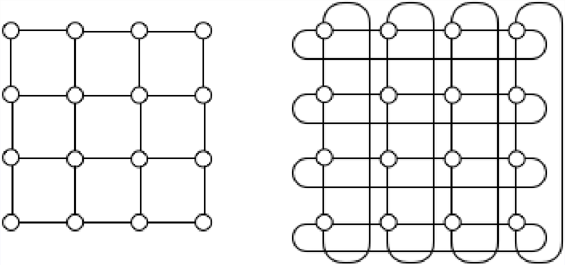
Definition.
Cartesian product of paths/cycles.

Coding of meshes (vertices and edges).

$$\{1,\ 2, \ldots,\ m\} \times \{1,\ 2, \ldots,\ n\}$$
$$\left\{\langle i,\ j\rangle \ \mid\ \left[i \in \{1,\ 2, \ldots,\ m\}\right],\ \left[j \in \{1,\ 2, \ldots,\ n\}\right]\right\}$$

Torus is obtained by adding the wraparound links...

Example for $n = 4$



Properties of the square torus with n vertices

$\sqrt{n}$ by $\sqrt{n}$

- Connected (diameter $D = \Theta(\sqrt{n}) = 2 \cdot \lfloor \frac{\sqrt{n}}{2} \rfloor$)
- Regular graph (degree $\delta = 4 = \Theta(1)$)
- Number of edges $2n = \Theta(n)$

Hypercubes

Motivation:
build a graph with a trade-off beetwen the degree and the diameter.

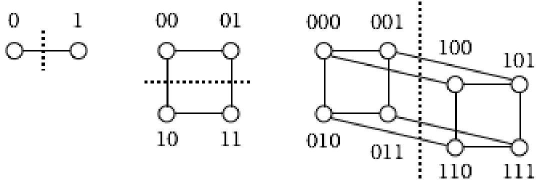
Hypercubes

Motivation:
build a graph with a trade-off beetwen the degree and the diameter.

Recursive Definition.

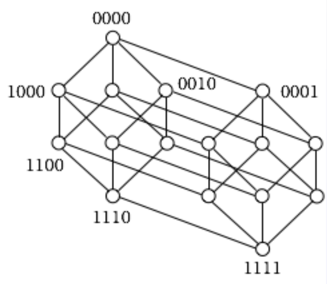
- The order-0 boolean hypercube, H_0, has a single vertex, and no edges.
- The order-$(k+1)$ boolean hypercube, H_{k+1}, is obtained by taking two copies of H_k ($H_k^{(1)}$ and $H_k^{(2)}$), and creating an edge that connects each vertex of $H_k^{(1)}$ with the corresponding vertex of $H_k^{(2)}$.

Construction



The next dimension

Representation of H_4



Coding

A natural binary coding

The coding from the vertices is naturally in the binary system.

The coding of two adjacent vertices is obtained by flipping only one bit.

Characteristics of Hypercubes

The number of vertices is a power of 2: $n = 2^k$ $(k = log_2(n))$

- Diameter $D_n = k$
- Degree $\delta_n = k$
- Number of edges?

Characteristics of Hypercubes

The number of vertices is a power of 2: $n = 2^k$ $(k = log_2(n))$

- Diameter $D_n = k$
- Degree $\delta_n = k$
- Number of edges?

  H_{k+1} is obtained by two copies of H_k plus 2^k edges for linking each relative vertex, thus:
  $N_{k+1} = 2 \times N_k + 2^k$ starting at $N_0 = 0$

  $N_k = k \times 2^{k-1}$

Graph isomorphism

The difficulty here is that there are many ways to draw a graph...

Property.

The hypercube H_4 is identical to the 4 by 4 torus.

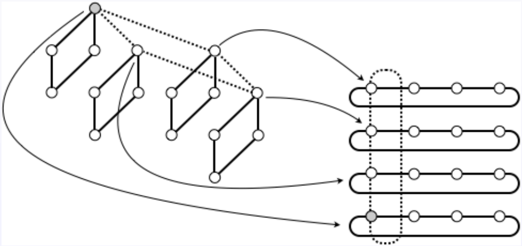
Graph isomorphism

The difficulty here is that there are many ways to draw a graph...

Property.

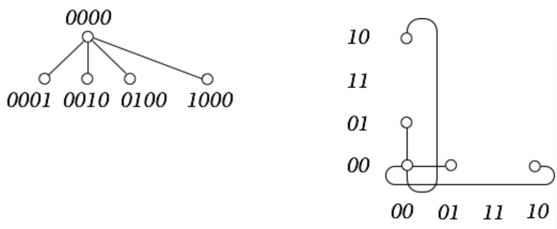
The hypercube H_4 is identical to the 4 by 4 torus.

The proof is by an adequate coding of the vertices/edges.

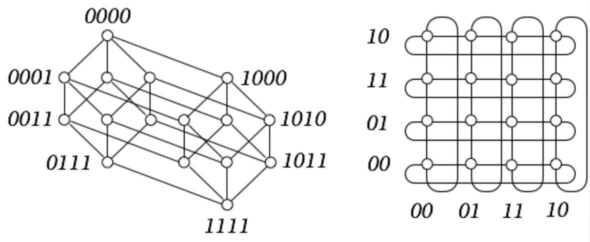


Coding schemes

The following figure (left) depicts this coding of a vertex and its neighbors.



The (almost) full picture



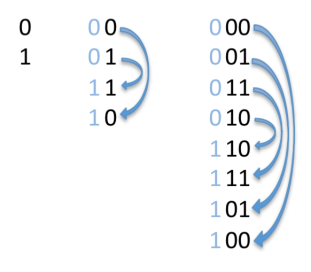
Gray Codes

Cultural aside

Let us present the most popular code, namely, the **Reflected Gray code**

The 1-bit Gray code is simply 0 and 1.
The next one (for 2-bits) is obtained by mirroring the 1-bit code and prefix it by 0 and 1.
The next ones are obtained similarly.

Gray Code



How to obtain the Gray code from the binary code?

Gray code can easily be determined from the classical binary representation as follows
$(x_{n-1}x_{n-2}...x_1x_0)_2$
shift right:
$(0x_{n-1}x_{n-2}...x_1)_2$

Take the exclusive OR (bit-to-bit) between the binary code and its shifted number:
$(x_{n-1}(x_{n-2} \oplus x_{n-1})...(x_0 \oplus x_1))_G$

For instance the binary code of $5 = (00101)_2$ is
$(0 \oplus 0)(0 \oplus 0)(0 \oplus 1)(1 \oplus 0)(0 \oplus 1) = (00111)_G$.

00001		00001
00010		00011
00011		00010
00100		00110
$\boxed{00101}$	$\longrightarrow$	$\boxed{00111}$
00110		00101
00111		00100
01000		01100
01001		01101
01010		01111
01011		01110
01100		01010
01101		01011
...		...

Matching

Definition

A matching is a set of edges that have no vertices in common.

It is *perfect* if its vertices are all belonging to an edge[1].

Proposition.

The number of perfect matchings in a graph of order $n = 2k$ grows exponentially with k.

[1] thus, the number of vertices is even and the cardinality of the matching is exactly half of this number

Example

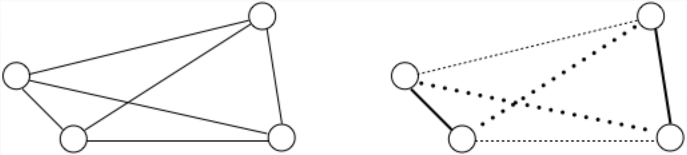


Figure: The 3 possible perfect matchings of K_4.

Proof

by recurrence on k,
let denote the number of perfect matchings by N_k.

Base case: For $k = 1$, there is only one perfect matching $N_1 = 1$ and for $k = 2$, there are 3 different perfect matchings $N_2 = 3$.
Induction step: For k, there are $2k - 1$ possibilities for a vertex to choose an edge, $N_k = (2k - 1).N_{k-1}$
Thus, N_k is the product of the k first odd numbers.

However, determining a perfect matching of minimal weight in a weighted graph can be obtained in polynomial time (using the Hungarian assignment algorithm).

Another interesting class of graphs.

Bipartite graphs.

A graph G is bipartite if its vertices can be partitioned into (by definition of partition, disjoint) sets X and Y in such a way that every edge of G has one endpoint in X and the other in Y.

An interesting question is to link bipartite graphs and matchings.

Trees

Definition

Trees are identified mathematically as graphs that contain no cycles or, equivalently, as graphs in which each pair of vertices is connected by a unique path.

A tree is thus the embodiment of “pure” connectivity, which provides the minimal interconnection structure (in number of edges) that provides paths that connect every pair of vertices.

Proposition

Any tree of order n has $n-1$ edges.

Example

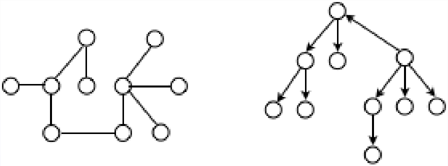


Figure: Undirected and directed trees.

Preliminary results

Lemma

1. Let G be a connected graph with $n \geq 2$ vertices.
   Every vertex of G has degree at least 1.
2. Any connected tree of order n ($n \geq 1$) has at least one vertex of degree 1 (called a leaf).

Preliminary results

Lemma

1. Let G be a connected graph with $n \geq 2$ vertices.
   Every vertex of G has degree at least 1.
2. Any connected tree of order n ($n \geq 1$) has at least one vertex of degree 1 (called a leaf).

Rapid Proofs

The main argument is on the analysis of graphs with minimum degrees 0 for part 1 and more than 2 for part 2.

Proof 1

Principle

By induction on the order of the graph n.

Proof 1

Principle

By induction on the order of the graph n.

Base case for $n = 2$
Induction step. Use the previous Lemma.

Proof 2

Inductive hypothesis. Assume that the indicated tally is correct for all trees having no more than k vertices.

Inductive extension. Consider a tree T with $k + 1$ vertices.

By the Lemma, T must contain at least one vertex v of degree 1.
If we remove v and its (single) incident edge, we now have a tree T' on k vertices.
By induction, T' has $k - 1$ edges. When we reattach vertex v to T', we restore T to its original state.
Because this restoration adds one vertex and one edge to T', T has $k + 1$ vertices and k edges.

Spanning Trees

Let consider a weighted graph G.

Motivation

a way of succinctly summarizing the connectivity structure inherent in undirected graphs.

Definition

Take the same set of vertices and extract a set of edges that spans the vertices.

Determining a minimal Spanning Tree is a polynomial problem. There exist two possible constructions, following two different philosophies.